Multi vehicle routing with nonholonomic constraints and dense dynamic obstacles

Mansouri, Masoumeh; Lagriffoul, Fabien; Pecora, Federico

DOI:
10.1109/IROS.2017.8206195

Multi Vehicle Routing with Nonholonomic Constraints and Dense Dynamic Obstacles

Masoumeh Mansouri[1] and Fabien Lagriffoul[1] and Federico Pecora[1]

***Abstract*— We introduce a variant of the multi-vehicle routing problem which accounts for nonholonomic constraints and dense, dynamic obstacles, called MVRP-DDO. The problem is strongly motivated by an industrial mining application. This paper illustrates how MVRP-DDO relates to other extensions of the vehicle routing problem. We provide an application-independent formulation of MVRP-DDO, as well as a concrete instantiation in a surface mining application. We propose a multi-abstraction search approach to compute an executable plan for the drilling operations of several machines in a very constrained environment. The approach is evaluated in terms of makespan and computation time, both of which are hard industrial requirements.**

I. Introduction

The Multi Vehicle Routing Problem (MVRP) [1] is a combinatorial optimization problem which consists of finding an optimal set of routes for a fleet of vehicles delivering goods to customers. The MVRP can be represented as a graph, where nodes are the locations (one for each customer), and the problem is to find a shortest closed path (tour) for all vehicles such that every node is visited by only one vehicle, exactly once. MVRP generalizes the well-known Traveling Salesperson Problem (TSP), that is, to find the shortest tour visiting every node in a graph exactly once. Strongly motivated from robotics, particularly the surveillance domain for UAVs, the TSP has also been adapted for vehicles with nonholonomic/dubins constraints. Two variants of this problem have been studied: the Euclidean TSP (ETSP), where the Euclidean metric is used to measure the distance between adjacent nodes; and the Dubins Traveling Salesperson Problem (DTSP) [2], where the problem is to find a shortest tour composed of paths of bounded curvature [3].

In a MVRP, a node associated to a vehicle should be traversed only once, and no other vehicle is allowed to traverse that point in their tour. Roughly speaking, each target along a vehicle's tour acts as an "obstacle" that appears dynamically once the node is visited, and which must be avoided by the tour of other vehicles. However, the MVRP employs the abstract notion of graph to represent locations and their connectivity, thus ignoring the spatial extent of the locations. The DTSP considers some physical characteristics of the environment (e.g., kinematic constraints of the vehicle), however these features are assumed to be static. To the best of our knowledge, no variants of the DTSP and MVRP that consider locations as emerging obstacles have been studied (see examples in [2], [4]). Also, the only variant of the MVRP that considers nonholonomic/dubins constraints [5] assumes a non-dense distribution of vehicle locations. As we will see, this assumption cannot be made in at least one important application domain.

In this paper, we introduce a variant of MVRP, called MVRP-DDO, in which we consider (1) nonholonomic/dubins constraints, (2) dynamic obstacles, and (3) dense vehicle locations. It is easy to see why the combination of these three factors makes MVRP-DDO a significantly different (and harder) problem compared to MVRP and DTSP. In the latter two, any decision on the sequencing of locations preserves the solvability of the problem (although not necessary the optimality of the solution). In MVRP-DDO, an obstacle emerges in a location when it is visited by a vehicle, and locations may be densely placed. Combined with non-trivial constraints on motion, this may lead to locations being unreachable, hence even solvability depends on location sequencing. MVRP-DDO is motivated by a surface mining application, which we explain in the following Section.

II. Motivating Industrial Application

Fig. 1. Two Atlas Copco drilling machines (Pitviper-351) in the process of drilling targets in a bench.

MVRP-DDO is motivated by a mining application, where a fleet of drilling machines operates on an open space (called bench) in an open-pit mine. A set of *drill targets* in the bench is given; at each target, a blast hole is to be drilled. The blast holes are then filled with explosive material that will be detonated after all targets have been drilled. After the explosion, the ore is taken away and processed for mineral extraction. The problem is to coordinate the motions of multiple drills operating concurrently on the bench.

For each drill target, a machine can autonomously carry out a set of tasks: auto-tramming (navigating to the target from its current position), leveling (deploying jacks for

[1]Center for Applied Autonomous Sensor Systems, Örebro University, SE-70182 Sweden. {mmi, fll, fpa}@aass.oru.se

horizontally leveling the machine), drilling, and de-leveling (retracting the jacks so the machine is placed back on its tracks). Each drilling machine has a square dust guard around its drill bit. The dust guard contains the pile of excess material produced by drilling the hole. This pile constitutes an obstacle for the machine that drilled the hole and for all other machines. One side of the dust guard can be lifted after drilling, which allows a machine to navigate to the next target without colliding with the pile that has accumulated under it after drilling. The distance between each target is approximately 9 meters, and the vehicle's base is a rectangle of size 16×12 m^2.

III. Related Work

The variants of the TSP that are relevant to our problem are the DTSP and the MVRP with nonholonomic constraints. Most existing algorithms for DTSP work based on a solution derived from ETSP (Euclidean TSP). In the ETSP, the kinematic constraints of the robot are usually not taken into account, whereas DTSP algorithms find optimal trajectories between locations, given the ordering obtained by solving a ETSP. When the minimum Euclidean distance between any two locations is large compared to the turning radius of the vehicle, the DTSP and ETSP are the same [6]. Conversely, if the locations are densely distributed in the plane, as is the case in MVRP-DDO, then algorithms based on the Euclidean metric are not necessarily a good choice, since many maneuvers are required. For these cases, angular-metric TSPs have been studied [7]. A variant of the TSP, where locations are regions instead of point locations, is called Traveling Salesperson Problem with Neighborhoods (DTSPN). However, the algorithms that have been studied for DTSPN do not consider these regions as obstacles for other vehicles [4], [8], therefore, DTSPN is not considered as an instance of MVRP-DDO.

Dynamic Vehicle Routing problems (DVR) [9] are online variants of TSP and MVRP, where locations to be reached become known only during execution. This makes the objective function hard to compute. In MVRP-DDO, the objective function is hard to compute despite the locations being known beforehand. MVRP-DDO is a hard offline problem, primarily because obstacle locations depend on robot allocation and order of missions.

A consistent body of research has focused on MVRP with nonholonomic constraints. Rathinam et al. [5] propose an algorithm for solving an instance of this problem under the assumption that no two locations are closer than twice the minimum turning radius. This is a very restricting assumption: in our mining application, for instance, the average distance between locations (drill targets) is much less than the length of the vehicle. The same type of restriction exists in other work [10], [11] where transformation techniques are employed to solve an Asymmetric TSP for solving the original MVRP for a fleet of heterogeneous UAVs. These approaches also rely on a procedure to efficiently calculate the cost of transitioning between any two locations, which is not trivial in MVRP-DDO due to the emergence of obstacles in locations that depend on the order of graph traversal.

Another drawback which is common to all approaches for MVRP with nonholonomic constraints mentioned above is that none of them consider vehicle collision avoidance. Considering collision avoidance between vehicles could be achieved by altering the estimate of travel cost on the basis of the positions of other vehicles, which would clearly levitate the computational overhead of cost estimation. Collision avoidance becomes crucial in small and/or dense environments when several vehicles move concurrently, and that is why our formulation of the MVRP-DDO (see Section IV) explicitly captures this requirement.

Motion planning with movable obstacles [12], as well as some rearrangement problems such as the Sokoban puzzle [13], are also relevant to MVRP-DDO. Similarly to MVRP-DDO, in these problems obstacles are dynamic. In particular, a robot can move obstacles around (e.g., by pushing), thus changing the state space of obstacles and free space. It has been shown that motion planning with movable obstacles is more complex than conventional motion planning [12], and even a simplified variant of this problem is NP-hard [14]. Sokoban is proved to be NP-hard and PSPACE-complete [15]. Because both MVRP-DDO and these problems share the characteristic of a dynamic state space, we suspect that MVRP-DDO is at least as complex[1] as these problems.

IV. Problem Definition

In this section, we define MVRP-DDO formally. A set of n vehicles is given, and let $T = \{\tau_1, \ldots, \tau_m\}$ be a set of *targets*. The coordinates (x_i, y_i) of each τ_i are given. Let $\prec$ be a partial order on set T, and let $(T, \prec)$ be the corresponding poset. We denote with $\tau_i \prec \tau_j$ the fact that τ_i precedes τ_j in the poset. Let H_T be the Hasse diagram of the poset, i.e., $(\tau_i, \tau_j) \in H_T$ iff $\tau_i \prec \tau_j$ and $\nexists \tau_k : \tau_i \prec \tau_k \prec \tau_j$. Given a poset $(T, \prec)$, and $A \subseteq T$, we refer to the least upper bound of A as *lub*, and the greatest lower bound of A as *glb*. We denote $len_{H_T}(A)$ the length of a path in the Hasse diagram H_T between the *lub* and the *glb*. Let $f(\boldsymbol{x}) = C$ be the kinematic model[2] of the vehicles, where $\boldsymbol{x}$ is the state of a vehicle. Let $P(i, j)$ be a kinematically feasible path between the poses (x_i, y_i, θ_i) and (x_j, y_j, θ_j), and let $len_P(i, j)$ denote the length of this path. We denote that a path $P(i, j)$ intersects a path $P(k, l)$ with the notation intersects(i, j, k, l), and that $P(i, j)$ intersects with the obstacle that emerges at the position of target τ_k with the notation intersects(i, j, k). Let $v_i = j$ denote the fact that vehicle i is assigned to reach target τ_j. We define $Q = \bigcup_{k=1}^{n} q_k$ to be an n-partition of T, where $q_k = \{\tau_i \in T : v_k = i\}$, that is, each partition q_k consists of the targets assigned to vehicle k. Given $q_k \in Q$, let H_{q_k} be the Hasse diagram of poset $(q, \prec)$, that is, H_{q_k} contains an edge (τ_i, τ_j) if vehicle k reaches target τ_j right after it has reached target τ_i without visiting any other

[1] A formal proof of problem complexity is the topic of future work. Note that MVRP-DDO may even be more difficult that these problems in practice, as it involves multiple robots.

[2] We assume that all vehicles have the same kinematic model.

target. Note that the Hasse diagram H_T may indicate that τ_j is preceded by τ_l, however v_l is necessarily different from k because $(\tau_i, \tau_j) \in H_{q_k}$.

Now, given n vehicles and a set of targets T, the problem is to determine θ_i and v_i for all $\tau_i \in T$ and the partial order $\prec$ such that the following objective function is minimized

$$\max_{q_k \in Q} \left\{ \eta_1 len_{H_T}(q_k) + \eta_2 \sum_{(\tau_i,\tau_j)\in H_{q_k}} len_P(i,j) \right\} \tag{1}$$

subject to the following constraints:

$$\forall q \in Q, (\tau_i, \tau_j) \in q^2 \text{ s.t. } i \neq j \;.\; \tau_i \prec \tau_j \vee \tau_j \prec \tau_i \tag{2}$$

$$\forall q \in Q, (\tau_i, \tau_j) \in q^2 \text{ s.t. } (\tau_i, \tau_j) \in H_q \;.\; \exists P(i,j) \tag{3}$$

$$\forall (\tau_i, \tau_j, \tau_k) \in T^3 \text{ s.t. intersects}(i,j,k) \;.\; \tau_j \prec \tau_k \tag{4}$$

$$\forall q_z, q_{w\neq z} \in Q, (\tau_i, \tau_j) \in H_{q_z}, (\tau_k, \tau_l) \in H_{q_w} \text{ s.t. intersects}(i,j,k,l) \;.\; \tau_j \prec \tau_k \vee \tau_l \prec \tau_i \tag{5}$$

$$f(\boldsymbol{x}) = C \tag{6}$$

The objective function (1) provides an indirect measure of the *makespan* of the solution: $len_{H_T}(q_k)$ counts the number of direct precedences that are imposed on vehicle k, while the summation over the edges of H_{q_k} measures the combined length of all paths connecting targets that vehicle k will visit. The constants η_1 and η_2 are normalization constants. The maximum over all assigned vehicles ultimately identifies the vehicle that travels the longest distance and has to yield to other vehicles more. Constraint (2) guarantees that the partial ordering is such that a sequence is assigned to each vehicle. Constraint (3) ensures that the partial ordering is such that there exists a feasible path between ordered targets. Constraint (4) imposes that paths intersecting any target occur before that target is reached (hence, before the target is covered by an obstacle). Constraint (5) guarantees that vehicle paths do not intersect with each other. Finally, paths also must satisfy kinematic constraints (6).

The MVRP-DDO can be understood as a combination of several sub-problems: deciding an allocation of vehicles to targets (robot allocation), deciding approach angles at which robots should place themselves on each allocated target, deciding the order of target traversal (sequencing), and computing feasible motions between subsequent targets (motion planning). Some of these sub-problems have been studied in combination, from which we can derive a partial understanding of the complexity of MVRP-DDO: MVRP has been shown to be NP-Hard [1], as has motion planning [16], and multi-robot scheduling is an NP-complete decision problem [17]. A previous attempt at systematically exploring the joint search-space of most sub-problems underlying the MVRP-DDO [18] reveal just how hard the problem is, and suggests that tackling sub-problems individually may be the only way to scale to realistically-sized MVRP-DDOs.

In addition to being hard, the sub-problems subsumed by MVRP-DDO are strongly dependent. For example, a solution to the MVRP sub-problem must also guarantee that the sequence of targets can be visited with kinematically feasible motions in the presence of emerging obstacles. The opposite dependency also exists, i.e., it may be impossible to sequence targets due to a particular choice of motions. For this reason, it is essential to consider the restrictions imposed by other sub-problems in solving each of them. In the following section, we propose an approach that explicitly considers relations between sub-problems, and also takes advantage of problem structure to identify easy choices early on.

V. Method

We provide a solution to an instantiation of the MVRP-DDO in a real surface mining application. Several algorithms are used in sequence to find kinematically-feasible and obstacle-free paths for a set of drilling machines that must cooperatively visit all drilling locations in a bench. As explained in Section II, piles are created as a result of drilling which constitute obstacles for machines navigating on the bench. In this MVRP-DDO, a good solution should minimize the time to completion of all navigating and drilling operations. Since all machines drill targets with similar efficiency, this measure is roughly proportional to the objective function (1). A desired final "parking" pose is given for each machine (see Figure 3). This enforces that the solution should be such that machines can reach their final parking pose without running over the piles.

In addition to the constraints (2)–(5), the paths in a solution of this MVRP-DDO must also be confined to a so-called geofence, a virtual fence within which it is safe for the vehicles to operate. A geological survey of the area and the production target of the mine determine the locations of drill targets and the geofence. The number of available machines depends on the size of the bench, and is assumed given.

Realistically-sized benches usually consist of hundreds of drill targets. Optimization problems of such scale often call for approximate solutions rather than exact ones. In order to solve this MVRP-DDO approximately, we decompose the problem into several sub-problems, and devise a hierarchical method to combine solutions of these sub-problems. The hierarchical method uses multi-abstraction search for minimizing the makespan. It is common that large-scale problems have an almost-decomposable structure, where the problem entities (in this case, the targets) can be clustered by one or more common properties. Our multi-abstraction search benefits from a pre-processing step, which clusters targets into meaningful groups. As we show, grouping reveals equivalence classes of sequencing and motion planning decisions, which significantly reduces the overall search space.

The MVRP-DDO is divided into the following sub-problems:

1) *Grouping:* divide targets into groups through pre-processing.
2) *Machine allocation:* allocate machines to targets given the available machines and their initial positions.
3) *Sequencing:* decide a sequence of targets for each machine.

4) *Approach angle:* decide a pose for each machine at each target.
5) *Motion planning:* decide how to navigate between poses given sequences and approach angles.
6) *Coordination:* schedule machines given kinematics and nominal speed of the vehicles.

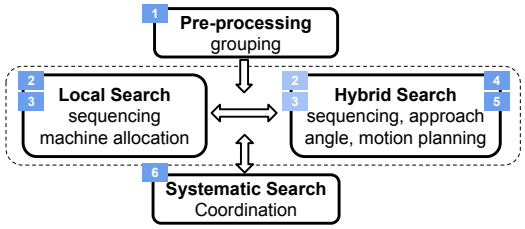

Fig. 2. An overview of multi-abstraction search for solving MVRP-DDO.

Figure 2 illustrates how the sub-problems above interact within an overall architecture. The pre-processing step creates an abstraction of the problem by clustering drill targets into groups. Groups contain targets that are "aligned", that is, they can be visited sequentially along paths with limited curvature. The pre-processing step is detailed below in Section V-A. The number and location of groups will be used by a local search method, namely, a Simulated Annealing combinatorial optimizer (see Section V-B). Local search contributes to solving the sequencing and the machine allocation sub-problems. The solution is an abstraction, as it does not prescribe specific paths to machines, nor does it concern itself with fine-grained scheduling decisions. The abstract solution is then refined by motion planning and a hybrid search method. The motion planning finds paths given the pile obstacles. The piles that should be considered by motion planning are computed from the sequencing decisions. Some approach angles are also decided as a direct consequence of the sequencing, namely those of targets that are far from the geofence. The approach angles of targets that are close to the geofence, henceforth called *hard targets*, are not trivially computed. We have developed a hybrid search to decide the approach angles and the sequencing of hard targets in an integrated manner (details are given in Section V-C). The last step in the refinement process is machine coordination, given the motion plans, sequencing, and machine assignment. A solution to the coordination sub-problem is a set of flexible temporal bounds within which the machines can carry out their motion, drilling, leveling and de-leveling operations. These bounds are computed by a scheduling algorithm, which guarantees that machines do not collide with other machines or with existing piles during their motions (see Section V-D).

The overall multi-abstraction search is inspired by optimizing multi-agent placement with task assignment and scheduling [19], where an abstract solution is refined incrementally with different types of search at different levels of abstraction. In the following, we explain the algorithm pipeline in detail.

A. Pre-processing

In an open-pit mine, drill targets usually lie on an irregular hexagonal grid (see Figure 3). We are interested in analyzing the topology of a bench in order to extract the principal directions of drill targets. This will allow to cluster targets into groups for which there are only few reasonable sequencing possibilities and that are easy to navigate in sequence.

A distance threshold is used to identify neighboring drill targets. A K-Means clustering of the set of angular coefficients of topologically neighboring drill targets discovers the principal directions. This yields clusters containing similarly oriented edges of the topology. These are used to group drill targets into roughly-parallel lines. There is usually more than one principle direction in a bench (e.g., see Figure 3 for possible different groupings). Among all the groupings, we select the one where the extracted lines are roughly perpendicular to the open area (e.g., grouping A in Figure 3). The reason is as follows: in grouping B, suppose that the first machine is allocated to drill the groups 1 to 3, and the second one is allocated to the groups 4 to 7. In the case of concurrent movement of these machine, either the first machine is locked in the bench due to the piles created by the second machine (thus not being able to reach its final parking pose), or the second machine should delay its operation in order to leave an "escape corridor" for the first machine. Neither of these problems would occur with grouping A, regardless of how the two machines are allocated to groups.

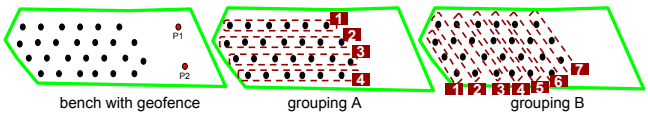

Fig. 3. A toy bench with drill targets (black circles), a geofence (green polygon), two parking positions; two different groupings are shown.

B. Local Search

The next step in our pipeline is solving the sequencing and the machine allocation sub-problems. This step considers how the decisions in these sub-problems affect the motion planning and the coordination sub-problems. We employ a Simulated Annealing algorithm which minimizes a lower bound of the makespan. Allocations and sequencing decisions are explored, subject to some of the constraints in MVRP-DDO. The sequencing of targets, constraint (2), is enforced at the level of groups of targets, and for the targets that are not hard. No motion planning is performed at this level, therefore, explicitly omitting constraint (3) in the local search. However, we use a rough estimation of the spatial occupancy of piles to impose an ordering among the groups, thus considering constraint (4) and (5) at a higher level of abstraction than that of individual targets.

A state in the local search represents an assignment of machines to groups, an ordering of groups, and an ordering among the targets in a group. We call this ordering the *direction* of a group, which can take one of two values: *upwards*, and *downwards*. The upward direction indicates that the sequence of targets explored by the machine should

start from the target that is furthest away from the geofence, and end at the one closest to the geofence (groups 1, 6, 7, 8 and 10 in Figure 4); the downward direction indicates the opposite (groups 2, 3, 4, 5, 9).

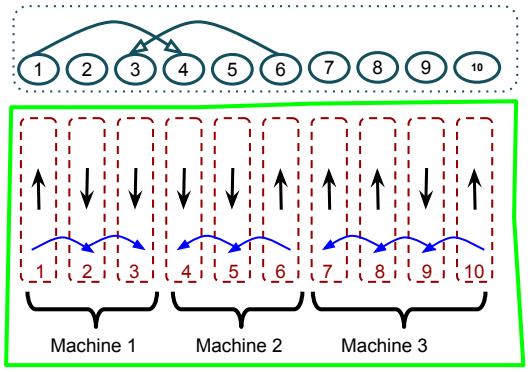

Fig. 4. A bench with grouped drill targets (red dashed rectangles) and a geofence (green polygon). Black arrows represent the direction in each group, blue arrows represent sequencing among the groups assigned to each machine. The resulting dependency graph G_d is shown above.

The neighborhood function used by the local search to generate a next state to explore randomly varies the machine allocations, the group sequences, and the group directions (see an example of a state in the Figure 4).

The cost function estimates the makespan of a state. Each group i is associated with two intervals: $I_h(i)$, representing when the machine assigned to group i occupies (navigating and drilling) an area containing the hard targets in the group; and $I_o(i)$ representing the time taken by the machine to traverse the area containing the other targets in the group. The time intervals associated to a group are approximations of the actual time it will take a machine to navigate and drill the targets, as this estimate is obtained in the absence of motion plans for transitioning between targets. We illustrate the reason for associating two distinct intervals to every group with an example. Consider the situation depicted in Figure 4: machine 1 starts drilling upwards towards the geofence; traversal over these targets most likely does not affect the motions of other machines, and the motions themselves are aligned to the principal direction of the group. However, when the machine navigates over the hard targets, finally transitioning from group 1 to group 2, its possibility to maneuver will be impeded by the densely placed piles it has created, as well as its proximity to the geofence. As we explain in Section V-C, group transitions require sequences that alternate drill targets in the two groups. More importantly, group transitions also tend to require motions that occupy more space. In the example, transitioning from group 1 to group 2 may require the machine to maneuver over an area that intersects the hard targets of groups 3 and 4. This is due to the confined space imposed by the geofence. The area associated to the hard targets in a group includes all of the group's hard targets, plus the hard targets of the three adjacent groups. The choice of three adjacent groups reflects experimental observation.

Function `cost-function` (s) : makespan

```
1  G_d ← Compute(N, E)
2  if FeasibilityEval(s, G_d) then
3      return ∞
4  makespan ← 0
5  if |E| = 0 then
6      for i : n do
7          t_v = Eval(s, v_i, ∅)
8          if t_v > makespan then
9              makespan ← t_v
10     return makespan
11 foreach R ∈ P(n, n) do
12     foreach v ∈ R do
13         t_v = Eval(s, v, G_d)
14         if t_v > makespan then
15             makespan ← t_v
16 return makespan
```

Given a state, $\texttt{cost}-\texttt{function}$ first constructs a dependency graph $G_d = (N, E)$ (line 1). The nodes of G_d represent the groups. An edge (i, j) exists in G_d if: (1) group i is upwards, hence the machine k traversing it is going towards the geofence; (2) its terminating targets are hard, hence the machine will navigate close to the geofence; (3) the group following group i is also assigned to machine k and is downwards, i.e., machine k will require complex maneuvers to leave group i; (4) group j is among the three adjacent groups to group i; (5) group j is assigned to another machine l. This situation implies that machine l cannot complete drilling the targets in group j because the space may be needed for machine k to complete a row transition. An example is shown in Figure 4.

The feasibility of the state is evaluated in line 2. A state in which both $(i, j) \in G_d$ and $(j, i) \in G_d$ is evaluated as being infeasible. If there is no dependency among the groups (line 5), i.e., the geofence is far from the groups, then, the `Eval` function estimates the makespan t_v for each machine v, given the current state, the Euclidean distance among the targets, a nominal speed, and a nominal drilling time. The maximum makespan among the machines is the output of the $\texttt{cost}-\texttt{function}$. If, on the other hand, there are dependencies in G_d, the `Eval` function considers the temporal overlaps of intervals associated to groups in calculating the makespan. For example (see Figure 4), when machine 1 switches from group 1 to group 2, it is considered to occupy the space of the hard targets in groups 2, 3 and 4 while performing its maneuvers. In this case, $(1, 4) \in G_d$ and therefore, the temporal overlap between $I_h(4)$ and $I_h(1)$ is summed to the makespan. Note that considering these temporal overlaps is very important for estimating the makespan, because machine 1 has to drill the hard targets between its maneuvers, and drilling time is much longer than driving time. Therefore, machine 2 has to wait for a very long time, and this time has to be reflected in the makespan. Since the order in which we compute the makespan of the machines matters in this computation, all the permutations of machines $R \in \mathscr{P}(n, n)$ are considered in computing the makespan (line 11).

C. Hybrid Search

Local search outputs a state where an allocation of machines to groups has been decided, as well as the direction of group traversal and the sequencing of groups. The sequence of non-hard targets within a group is directly determined from the direction of group traversal, and motions between each pair of targets in this sequence is easily computed via a motion planner (see below). Conversely, the sequencing of hard targets remains to be determined. Determining such sequences is not trivial because of the constraints imposed by previously drilled holes, and by the proximity of the geofence. Consider the example in Figure 5, where all targets are to be drilled by the same machine. The bottom group (G1) has been drilled (hence each of its targets acts as an obstacle), the middle group (G2) is about to be completed, and G3 is to be drilled next. However, the geofence (green) prevents the machine from performing a U-turn between targets 5 and 4, therefore a more complex sequence needs to be computed. The actual sequence computed for this example is given by the numbers in the figure.

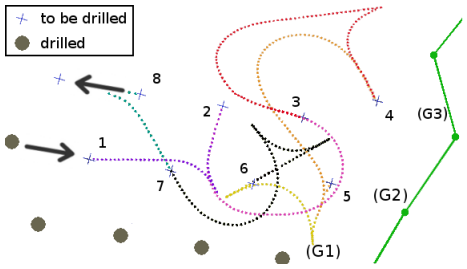

Fig. 5. Example of solution computed by the hybrid search algorithm. The targets are drilled following the sequence indicated by the numbers.

For each sequenced pair of targets (τ_i, τ_j), a kinematically feasible path (i.e., satisfying constraints (3) and (6)) is computed via a motion planner for a car-like mobile robot based on cubic spirals [20]. The motion planner accounts for the obstacles that are known to have emerged from the targets $\tau_k : \tau_k \prec \tau_j$ that have been drilled by the same machine. This is because only these obstacles are known to have appeared, whereas the precedences between τ_j and the targets drilled by other machines are not known. These will be revealed by coordination (see next Section), which adds precedences based on precise travel times of machines along the decided paths.

For the hard targets, sequencing and paths are computed jointly via a hybrid search. In particular, the hybrid search algorithm interleaves search between a discrete search space (the space of possible target sequences) and a geometric search space (the space of possible approach angles θ_i for the targets). The combined search space grows very quickly with the number of targets: for n targets, there are $n!$ possible drill sequences, and given k possible approach angles, this results in k^n ways of connecting the n targets. Moreover, motion planning needs to be performed to validate/reject each possible path.

Hybrid search needs to compute answers to these subproblems within a few seconds, as a bench typically includes tens, if not hundreds, of hard targets. The hybrid search algorithm tackles the complexity of the problem using two key insights:

- backward search in the space of drill sequences;
- incomplete search in the space of approach angles.

The first idea uses the fail-first principle: forward search often leads to situations where $n-1$ targets have been drilled, but the last target is no longer reachable, which leads to tremendous amount of backtracking. Backward search allows us to quickly prune out these dead-ends. The second idea consists in searching among the possible approach angles in a forward manner, but without backtracking, i.e., if no path is found after trying the k possible approach angles for the current target, the whole sequence is discarded, and another sequence is considered. The loss of completeness at the geometric level is compensated by the fact that many sequences are actually feasible.

D. Coordination

The last step in the refinement process is coordination. In the previous section we have shown how the motion planner ensures that no machine will collide with piles caused by itself (partially enforcing constraint (4)). However, possible collisions of a machine's path with the piles created by other machines have not been considered. Whether these collisions can occur depends on the time interval within which piles are created, therefore, a fine-grained representation of time and space is needed to account for identifying and removing these potential conflicts.

We employ a spatio-temporal representation of trajectories, called trajectory envelopes [21]. A trajectory envelope is a set of constraints that curtails the possible poses of a vehicle, and the times at which the vehicle can be in these poses. The piles are also represented by trajectory envelopes, which consist of one polygonal spatial constraint and a time interval that starts when the target is drilled and lasts forever. A precedence $\tau_j \prec \tau_k$ should be decided if the obstacle emerging from τ_k intersects $P(i,j)$ in space and its temporal interval intersects that of the trajectory envelope $P(i,j)$ (see constraint (4)). Similarly, two trajectory envelopes $P(i,j)$ and $P(k,l)$ should be sequenced if they intersect spatially and temporally (i.e., one of $\tau_j \prec \tau_k$ or $\tau_l \prec \tau_i$ should be decided, see constraint (5)). The set of precedences that needs to be imposed is decided by the Earliest Start Time Approach precedence-constraint posting algorithm for vehicle coordination [21]. This is essentially a backtracking search in the space of precedence constraints between spatio-temporally overlapping trajectory envelopes.

VI. Experiments

Automated fleet management solutions for the mining industry need to satisfy two requirements. First, drill plans should have a short makespan; computation times should be in the order of hours rather than days.

Short makespan can be understood as how close the makespan obtained by our approach is to the makespan of

plans designed by humans (which is the current practice in the industry). Such a comparative analysis is not feasible, as the disclosure of sufficiently many human-generated plans by mining companies is problematic. In order to explore a spectrum of easy and hard problems, we generate a benchmark of artificial problems where density and closeness of targets to the geofence is varied. We then compare the makespan of the solutions obtained by our algorithm with a lower bound on makespan that can be computed for each problem in the benchmark.

Our benchmark consists of 400 problems, all with 150 drill targets organized in an irregular hexagonal grid. These problems are divided into four types (100 problems for each type); each problem type is a tuple $\langle(\mu_r,\sigma_r),(\mu_g,\sigma_g)\rangle$, where (μ_r,σ_r) is the mean and standard deviation of the Normal distribution of radii of the hexagons. The drill targets are placed on the vertices of these hexagons. (μ_g,σ_g) is the mean and standard deviation of the distance between the geofence and each drill target on the external border of the grid. The problems of type 1 are $\langle(9.0,0.5),(9.0,0.5)\rangle$, i.e., drill targets are densely placed, and the geofence is very close to the drill targets; problems of type 2 are $\langle(9.0,0.5),(36.0,0.5)\rangle$, i.e., the drill targets are dense, but the geofence is far from the targets; problems of type 3 are $\langle(14.0,0.5),(9.0,0.5)\rangle$, i.e., the geofence is close to the targets, but the targets are far from each other; and problems of type 4 are $\langle(14.0,0.5),(36.0,0.5)\rangle$, i.e., the geofence is far from the targets, and the targets are sparse[3].

In order to evaluate the quality of solutions, we compute a lower bound (LB) of the makespan for each problem in all problem types. The LB is computed assuming straight-line movements between targets, a fair assignment of targets between machines, the shortest sequencing of targets with respect to the Euclidean distance, and a constant time for drilling. The first row of Table I shows the average percentage deviation of makespan obtained by our approach with respect to the LB. In the most constrained problems, our approach yields only a 13.24% higher makespan than the LB. Notice that in reality, machines need to perform many maneuvers to reach targets, and they also may need to delay their operations if the space they would use to maneuver is required by other machines. As expected, the deviation from the LB decreases as problems become less constrained (e.g., 1.68% increase in type 4).

All problems in the benchmark were solved by invoking each solver in the pipeline once. As we have discussed, the approach is incomplete. Failure to find a solution can be used to re-invoke the pipeline of solvers with new information (see below). In order to verify how effective the `cost-function` is for abstracting the spatio-temporal characteristics of the problem, we measure the success rate of one invocation of the pipeline. We solve all four types of problems with and without constructing G_d in the `cost-function`. The second row of Table I shows the percentage of solved problems. The result shows that a spatio-temporally rich `cost-function` that employs G_d significantly outperforms not considering G_d with regards to solvability. This also emphasizes that the source of difficulty is a combination of closeness of the geofence to the targets and target density, as the success rate in problems of type 1 is low.

We can also observe the quality of solutions with and without using G_d, as shown in Figure 6. Makespans of solutions to problems of type 4 (which are considered easy, since both the geofence and the targets are far from each other) are not affected, while those of hard problems (type 2) are.

TABLE I

SOLUTION QUALITY AND SOLVABILITY IN FOUR TYPES OF PROBLEMS.

Comparison	*type 1*	*type 2*	*type 3*	*type 4*
Makespan vs. LB	13.24%	8.56%	2.30%	1.68%
w/ G_d vs. w/o G_d	70:0 %	90:60%	100:3%	100:100%

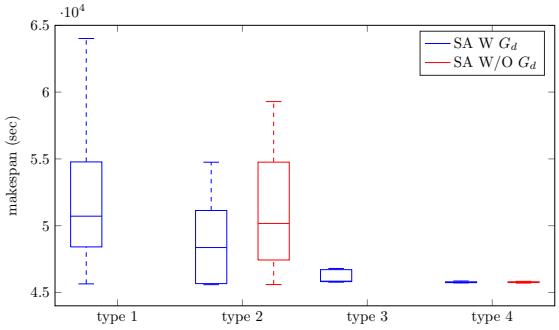

Fig. 6. Boxplot of the makespan for four types of problems, with and without G_d.

In order to evaluate the computation time (CT) of the overall multi abstraction process, we generate a further benchmark of 360 random problems of type 1 (hard problems) and 360 problems of type 4 (easy problems). In each type, there are 40 problems per problem size. Problem size varies from 100 targets to 900 targets by step increments of 100. We expect to see exponential growth of CT, as coordination is exponential in problem size due to the backtracking nature of the algorithm. In practice, however, we observe polynomial growth for both types of problems. A weighted least squares method for curve fitting finds a cubic polynomial fit as a best fit for the easy problems (with the root mean square (RMS) error equal to 0.2), as well as for the hard problems (with the RMS error equal to 0.09)[4].

These results are explained as follows. For easy problems, no backtracking was necessary during coordination. For hard problems, we show in Table II (row 1) the percentage of problems that required backtracking for various problem sizes. The second row shows the average number of backtracks per problem. What we understand from this result is that due to the spatio-temporal aware `cost-function` used in the local search, the need for coordinating machines is

[3]The video attachment shows solutions to smaller problem instances in the first and last category.

[4]The RMS error of an exponential fit for hard problems is 0.2.

reduced, which in turn, leads to a lower number of backtracks and milder growth rate in CT. We also report the average computation time used for motion planning and sequencing via hybrid search (row 3). As shown, this value is very low, as each group transition search involves few targets (only those that a particular machine uses to transition between two groups), and thanks to the most-constrained first principle.

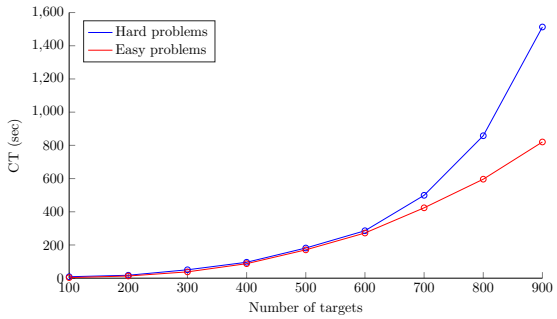

Fig. 7. Computation time vs. problem size.

TABLE II

BACKTRACKING AND HYBRID SEARCH TIMES FOR HARD PROBLEMS.

	100	*200*	*300*	*400*	*500*	*600*	*700*	*800*	*900*
BT%	67	20	20	10	15	10	20	32	20
BT AVG	11.8	2.55	1.75	0.57	1.2	0.7	0.9	0.7	0.8
HS (s)	0.09	0.12	0.13	0.15	0.17	0.19	0.22	0.21	0.19

VII. CONCLUSION

In this paper, we have introduced the MVRP-DDO, a problem which accounts for many features that are abstracted away in similar problems reported in the literature. MVRP-DDO is motivated by a real industrial application, and can be understood as a composition of tightly dependent sub-problems. We have also presented an instantiation of the problem in a mining application, and described a method for solving it via a combination of algorithms. These span levels of abstraction and deal with complementary aspects of the overall problem. We have shown how we can achieve high scalability and high solution quality (both of which are hard industrial requirements) by considering aspects of other sub-problems in each algorithm — e.g., considering time and space in the high-level local search, or exploiting the structure of the problem to identify easy choices.

MVRP-DDO in general is relevant to other application domains. These include multi-robot wheat harvesting [22], where machines must avoid areas that are already harvested. MVRP-DDO captures problems where the area covered by obstacles depends dynamically on the actions performed by robots. Other application domains where this is the case can be envisaged, e.g., multi-robot mopping or painting.

As suggested in Section VI, the `cost-function` can be learned incrementally from failures. An iterative version of the "one-shot" method used here would then explore many high-level solutions considering feedback obtained from the refinement process. This is the topic of ongoing work.

Acknowledgments. This work is supported by the Swedish Knowledge Foundation (KKS) project "Semantic Robots" and Atlas Copco Rock Drills AB. We are grateful to Pontus Bergsten and Robert Lundh for their support.

REFERENCES

[1] P. Toth and D. Vigo, Eds., *The Vehicle Routing Problem*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2001.

[2] K. Savla, E. Frazzoli, and F. Bullo, "On the point-to-point and traveling salesperson problems for Dubins' vehicles," in *Proc. of the American Control Conference*, vol. 2, Portland, OR, June 2005, pp. 786–791.

[3] L. E. Dubbins, "On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal position and tangents," *American J. Mathematics*, vol. 79, pp. 497–516, 1957.

[4] P. Váňa and J. Faigl, "On the dubins traveling salesman problem with neighborhoods," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 4029–4034.

[5] S. Rathinam, R. Sengupta, and S. Darbha, "A resource allocation algorithm for multivehicle systems with nonholonomic constraints," *IEEE Transactions on Automation Science and Engineering*, vol. 4, no. 1, pp. 98–104, 2007.

[6] J. Le Ny, E. Frazzoli, and E. Feron, "The Curvature-Constrained Traveling Salesman Problem For High Point Densities," in *Proceedings of the 46th IEEE Conference on Decision and Control*, 2007.

[7] A. Aggarwal, D. Coppersmith, S. Khanna, R. Motwani, and B. Schieber, "The angular-metric traveling salesman problem," in *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '97. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 1997, pp. 221–229.

[8] X. Zhang, J. Chen, B. Xin, and Z. Peng, "A memetic algorithm for path planning of curvature-constrained {UAVs} performing surveillance of multiple ground targets," *Chinese Journal of Aeronautics*, vol. 27, no. 3, pp. 622 – 633, 2014.

[9] F. Bullo, E. Frazzoli, M. Pavone, K. Savla, and S. L. Smith, "Dynamic vehicle routing for robotic systems," *Proceedings of the IEEE*, vol. 99, no. 9, pp. 1482–1504, 2011.

[10] S. G. Manyam, S. Rathinam, and S. Darbha, "Computation of lower bounds for a multiple depot, multiple vehicle routing problem with motion constraints," in *Proceedings of the 52nd IEEE Conference on Decision and Control, CDC 2013, December 10-13, 2013, Firenze, Italy*, 2013, pp. 2378–2383.

[11] P. Oberlin, S. Rathinam, and S. Darbha, "Today's traveling salesman problem," *IEEE robotics & automation magazine*, vol. 17, no. 4, pp. 70–77, 2010.

[12] M. Stilman and J. J. Kuffner, "Navigation among movable obstacles: Real-time reasoning in complex environments," *International Journal of Humanoid Robotics*, vol. 2, no. 04, pp. 479–503, 2005.

[13] A. Junghanns and J. Schaeffer, "Sokoban: A challenging single-agent search problem," in *In IJCAI Workshop on Using Games as an Experimental Testbed for AI Reasearch*, 1997.

[14] G. Wilfong, "Motion planning in the presence of movable obstacles," in *Proceedings of the Fourth Annual Symposium on Computational Geometry*, ser. SCG '88. ACM, 1988, pp. 279–288.

[15] J. Culberson, "Sokoban is pspace-complete," *Technical Report TR 97-02, Dept. of computing scinence, University of Alberta*, 1997.

[16] S. M. LaValle, *Planning Algorithms*. New York, NY, USA: Cambridge University Press, 2006.

[17] P. Brucker, *Scheduling algorithms*. Springer, 2004.

[18] M. Mansouri, H. Andreasson, and F. Pecora, "Hybrid reasoning for multi-robot drill planning in open-pit mines," *Acta Polytechnica*, vol. 56, no. 1, pp. 47–56, 2016.

[19] C. Zhang and J. A. Shah, "Co-optimizating multi-agent placement with task assignment and scheduling," in *Proc. of Inter. Joint Conference on Artificial Intelligence (IJCAI-16)*, 2016.

[20] T.-C. Liang, J.-S. Liu, G.-T. Hung, and Y.-Z. Chang, "Practical and flexible path planning for car-like mobile robot using maximal-curvature cubic spiral." *Robotics and Autonomous Systems*, vol. 52, no. 4, pp. 312–335, 2005.

[21] F. Pecora, M. Cirillo, and D. Dimitrov, "On mission-dependent coordination of multiple vehicles under spatial and temporal constraints," in *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2012.

[22] S. Scheuren, S. Stiene, R. Hartanto, J. Hertzberg, and M. Reinecke, "Spatio-temporally constrained planning for cooperative vehicles in a harvesting scenario," *KI*, pp. 341–346, 2013.